CFD – Mathematical basics

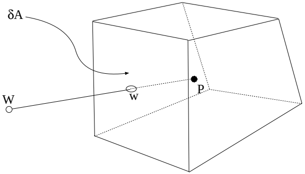
SOE3213/4: CFD Lecture 2

September 26, 2006

3d Transport Equation using FV method

$$\frac{\partial \phi}{\partial t} + \nabla.(\mathbf{u}\phi) = \nu \nabla^2 \phi + S_\phi$$

Divide region of interest into cells, integrate over each cell.



1st term

$$\iiint_{\delta V} \frac{\partial \phi}{\partial t}\, dV = \frac{d}{dt}(\phi \delta V) = \frac{\phi^{t+\delta t} - \phi^t}{\delta t} \delta V$$

2nd term

– convection of fluid into/out of cell

$$\iiint_{\delta V} \nabla.(\mathbf{u}\phi) = \sum \phi\mathbf{u}.\delta\mathbf{A}$$

e.g.

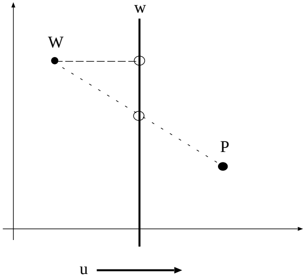
$$\sum \phi\mathbf{u}.\delta\mathbf{A} = (\phi \times u_x A)_w - (\phi \times u_x A)_e$$

CFD mesh can be

- colocated
  - all variables stored at cell centres
  - need to *interpolate*
- staggered
  - fluxes stored at cell faces
  - still need to work out fluxes by interpolation

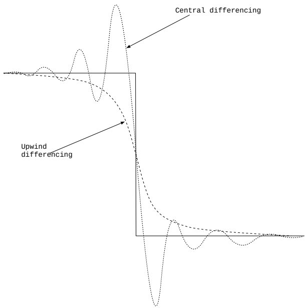
2 ways to get face value from cell centres

1. Value from upwind cell centre – upwind differencing
2. Average of the cell centres – central differencing



Often referred to as *differencing schemes*

1. Different errors :
   - Upwind differencing introduces *numerical viscosity*.
   - Central differencing introduces *phase errors*.



2. Also more sophisticated – blended differencing, higher order. . .

Mesh Generation

Advantage of FV method – cells can be any shapes necessary. However, numerical errors can be caused by bad meshes.

- Cells usually cubic (hexahedral) or tetrahedral
- (square or triangular in 2d)
- Tets (triangles) generated by automatic mesh generation (Fluent)
- Hex/square probably better numerically
- Cells should not be
  - long and thin
  - faces at odd angles
  - widely varying sizes

Matrix inversion

Diffusion & source terms also discretised similarly. Net result – numerical scheme (implicit)

$$\mathcal{M}[\phi^{t+\delta t}] = Q$$

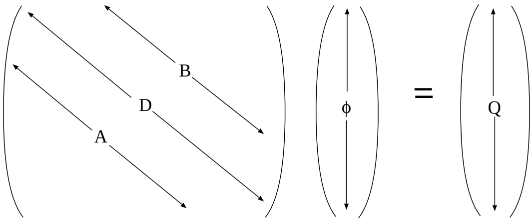
Solution step involves evaluating $\mathcal{M}^{-1}$ to advance solution by 1 (time)step.

In theory any valid technique could be used to invert $\mathcal{M}$. In practice, $\mathcal{M}$ is huge – numerical techniques need to be carefully constructed.

A good matrix algorithm needs to be

- Fast – should scale as N or $N \log N$
- Efficient – only store non-zero elements of the matrix

Most matrices are *sparse*. Eg. 1d FV method gives :



Only the elements D, A and B need to be stored in memory.

Navier-Stokes equations

$$\nabla.\mathbf{u} = 0$$

$$\frac{\partial u_x}{\partial t} + \nabla.(\mathbf{u}u_x) = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\nabla^2 u_x$$

$$\frac{\partial u_y}{\partial t} + \nabla.(\mathbf{u}u_y) = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\nabla^2 u_y$$

$$\frac{\partial u_z}{\partial t} + \nabla.(\mathbf{u}u_z) = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu\nabla^2 u_z$$

- Incompressible form
- Cons. of mass, momentum
- Linked – u_x, u_y, u_z, p all appear
- Non-linear – $\mathbf{u}u_x$